VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Outdoorová hra pro platformu iOS

Outdoor Game for iOS Platform

2015 Adam Zikmund

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Adam Zikmund**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Outdoorová hra pro platformu iOS
Outdoor Game for iOS Platform

Zásady pro vypracování:

Cílem bakalářské práce bude vývoj hry pro více hráčů pro platformu iOS s využitím geolokace a herního serveru, umožňujícího výměnu údajů mezi hráči. Hra bude využívat existující herní framework a bude založena na dříve řešené hře "Catch the Phantom". Bude možno definovat různé úrovně obtížnosti a komplexnosti konkrétní hry.

Pokyny k vypracování:
1. Nastudujte si problematiku tvorby her na mobilních zařízeních a problematiku her, využívajících určování polohy.
2. Navrhněte vhodný způsob komunikace se serverem. Definujte způsob autentizace a podobu API serveru a popište použitý komunikační protokol.
3. Hru implementujte na mobilní platformě iOS ve verzi 7 a vyšší, herní logiku oddělte od GUI.
4. Otestujte hru simulací pohybu hráčů a bude-li to možné, i v reálném provozu.

Seznam doporučené odborné literatury:

[1] Jonathan Zdziarski, iPhone SDK Application Development: Building Applications for the AppStore, O'Reilly Media; 1 edition, 2009, ISBN 0596154054
[2] Jonathon Manning, Paris Buttfield-Addison: IOS Game Development Cookbook. O'Reilly Media, Inc., 2014. ISBN 978-1-44936-876-0
[3] Dmitry Volevodz: iOS 7 Game Development. Packt Publishing Ltd., 2014. ISBN 978-1-78355-157-6.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Pavel Moravec, Ph.D.**

Datum zadání: 01.09.2014
Datum odevzdání: 07.05.2015

VYSOKÁ ŠKOLA BÁŇSKÁ - TECHNICKÁ UNIVERZITA OSTRAVA * Fakulta elektrotechniky a informatiky

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Souhlasím se zveřejněním této bakalářské práce dle požadavků čl. 26, odst. 9 Studijního a zkušebního řádu pro studium v bakalářských programech VŠB-TU Ostrava.

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 28. 4. 2015 .....

Rád bych na tomto místě poděkoval, především panu doktoru Pavlu Moravcovi za vedení bakalářské práce. Dále bych chtěl poděkovat panu doktoru Davidu Ježkovi za užitečné rady při tvorbě praktické části.

Abstrakt

Cílem této práce je implementace hry pro platformu iOS. Práce je zaměřena na popis principu herní logiky a také na řešení jednotlivých funkčních celků. Pro analýzu a návrh byly použity diagramy jazyka UML a pro implementaci objektově orientované jazyky Objective-C a Java. V práci se podařilo vytvořit funkční řešení pro server i klienta. Přínosem práce je, způsob jakým lze vytvořit hru pro platformu iOS postavenou na současných moderních technologiích.

Klíčová slova: mobilní hra, iOS, REST, geolokace

Abstract

The main goal of the thesis is the implementation of a game for the iOS platform. The thesis is focused on the description of game logic and example solutions for individual functional units. UML diagrams have been used for analysis and design, the implementation has been done using Objective-C and Java object-oriented languages. We have succeeded to create working server and client parts of the application. The thesis also describes how to implement a game for the iOS platform, which is built on today's modern technologies.

Keywords: mobile game, iOS, REST, geolocation

Seznam použitých zkratek a symbolů

UML	–	Unified Modeling Language
SDK	–	Software Development Kit
OS	–	Operating System
Wi-Fi	–	Wireless Fidelity
GSM	–	Global System for Mobile communications
SMS	–	Short Message Service
GPS	–	Global Positioning System
IBM	–	International Business Machines
ANSI	–	American National Standards Institute
ISO	–	lnternational Organization for Standardization
HTTP	–	HyperText Transfer Protocol
RPC	–	Remote Procedure Call
SOAP	–	Simple Object Access Protocol
XML	–	eXtensible Markup Language
DOM	–	Document Object Model
SQL	–	Structured Query Language
REST	–	REpresentational State Transfer
JSON	–	JavaScript Object Notation
UDID	–	Unique Device Identifier
NFC	–	Near field communication
IP	–	Internet Protocol
MVC	–	Model View Controller
API	–	Application Programming Interface
JPA	–	Java Persistence API
DTO	–	Data transfer object

Obsah

Seznam tabulek

Seznam obrázků

Seznam výpisů zdrojového kódu

1 Úvod

Uvedení první generace mobilního telefonu iPhone od společnosti Apple, přineslo světu nový standart pro chytrá zařízení. V době před uvedením na trh sice existovala zařízení, která uměla více než jen telefonovat a posílat SMS zprávy, ale často byla špatně ovladatelná a mnohdy i nefunkční. A právě snaha prvního iPhone byla, vyřešit všechny tyto nedostatky.

Trend chytrých zařízení tak doznal raketového růstu. Na trhu se začala objevovat nová chytrá zařízení a výrobci se snažili předčit konkurenci výkonnějšími komponentami či modernějšími funkcemi. V neposlední řadě chytrá zařízení ovlivnila každodenní život velkého množství lidí.

Motivací pro tuto práci je právě rozšířenost mobilních zařízení mezi běžnými lidmi. Díky tomu má vývojář možnost pomocí své aplikace zaujmout velký počet lidí bez toho, aby musel obtížně řešit distribuci. Dalším důvodem je skutečnost, že trh je plný nových technologií a získání praxe s těmito technologiemi je velmi důležitý pro budoucí uplatnění na trhu práce.

Práce si dává za cíl, vytvoření hry pro chytrá zařízení na platformě iOS. Princip samotné hry vychází z deskové hry Fantom staré Prahy. K přenesení principů z deskové hry na chytrá zařízení je nutno využití geolokačních služeb a také internetu pro komunikaci mezi serverem a klientem. Dále je kladen důraz na použití moderních technologií, které jsou v současné době dostupné na trhu. Právě použití nejnovějších technologií při návrhu a následné implementaci je stěžejním bodem této práce.

V první kapitole se práce zabývá historií a vývojem operačního systému iOS. V kapitole je dále seznam jednotlivých verzí iOS a také tabulka s přehledem podílu na trhu. Dále jsou popsaná jednotlivá zařízení, která podporují tento operační systém. V další kapitole je popis dostupných aplikací a her, které využívají geolokační služby. Třetí kapitola se věnuje popisu a historii jednotlivých technologií, které jsou použity při implementaci zadání. U každé z technologií je uvedena i její praktická ukázka. Popisu a vysvětlení herních principů a pravidel je věnována čtvrtá kapitola. V páté kapitole je podrobně popsán proces analýzy a návrhu klientské i serverové části. V případě serverové části se práce zabývá perzistentní vrstvou, komunikačním rozhraním a také herní logikou. U klientské části je vysvětlen postup pro návrh uživatelského rozhraní a určování polohy. Další kapitola podrobně popisuje samotnou implementaci obou částí a také poukazuje na problémy, které pří implementaci nastaly. Osmá kapitola se zaobírá testováním a zkušební hrou uskutečněnou v reálném prostředí. V poslední kapitole je popsán postup nasazení serverové části a proces jaký je potřeba uskutečnit pro úspěšné schválení a nasazení klienta do App Store.

2 Operační systém iOS

2.1 Představení operačního systému iOS

Poprvé byl operační systém představen 29. června 2007 společností Apple. Spolu s tímto systémem bylo představeno i první zařízení, na kterém tento systém běžel. Operační systém nenesl při představení žádné označení. Operační systém byl pojmenován až 6. března 2008, kdy byl uvolněn vývojářský balík iPhone SDK. Právě podle tohoto názvu byl operační systém poprvé pojmenován na iPhone OS. Poslední změna názvu operačního systému proběhla 7. června 2010, kdy byl název změněn na iOS.

iOS je výhradně určen pouze pro zařízení od společnosti Apple a tak jej nalezneme jen na produktových řadách iPhone, iPad a iPod. Díky tomu je tento systém distribuován pouze na zařízeních od společnosti Apple. Přesto patří mezi nejrozšířenější operační systémy pro mobilní zařízení na světě. Podle posledních údajů z roku 2014, které jsou zobrazeny v následující tabulce, patří právě iOS zhruba 20 % podíl.

Rok	Android	iOS	Windows Phone	Black Berry OS	Ostatní
2014	76,6%	19,7%	2,8%	0,4%	0,5%
2013	78,2%	17,5%	3,0%	0,6%	0,8%
2012	70,4%	20,9%	2,6%	3,2%	2,9%
2011	52,8%	23,0%	1,5%	8,1%	14,6%

Tabulka 1: Procentuální podíl operačních systému na trhu [1]

K rozšířenosti iOS hlavně dopomohlo silné postavení Applu s mobilními zařízeními na trhu a také to, že systém podporoval instalaci aplikací třetích stran pomocí služby App Store. Právě díky podpoře aplikací od třetích stran, se iOS dostal do povědomí vývojářů, kteří byli schopni vytvořit již přes 1,4 miliónu aplikací [2]. Další nespornou výhodou, která dopomohla k rozšířenosti operačního systému iOS jsou jeho pravidelné aktualizace. Apple každoročně představuje novou verzi systému, která obsahuje nové funkce. Přehled jednotlivých verzí systému iOS s výčtem nových funkcí můžete vidět v následující tabulce.

Datum představení	Verze	Nové funkce
29. červen 2007	1.0	Multi-touch gesta, mapy
11. červenec 2008	iPhone OS 2.0	App Store, aplikace třetích stran
17. červen 2009	iPhone OS 3.0	Push notifications, hlasové ovládání, Find My iPhone
21. červen 2010	iOS 4	Multitasking, složky, FaceTime
6. červen 2011	iOS 5	Siri, iMessage, Wi-Fi synchronizace
11. červen 2012	iOS 6	Apple Maps, integrace Facebooku
10. červen 2013	iOS 7	Control center, AirDrop
2. červen 2014	iOS 8	HealthKit, Continuity, Family Sharing

Tabulka 2: Přehled verzí operačního systému iOS

V roce 2015 je očekáváno představení nové verze iOS 9. Přibližná doba představení se odhaduje na přelom jara a léta.

2.2 Mobilní zařízení s operačním systémem iOS

2.2.1 iPhone

První iPhone představil Steve Jobs 9. ledna 2007 na Macworld konferenci. Při této události představil světu revoluční mobilní zařízení, které obsahovalo funkce telefonu, fotoaparátu, internetového komunikátoru a přehrávače hudby iPod. Představením prvního iPhone, Apple definoval představu, jakým směrem by se měla ubírat všechna chytrá mobilní zařízení. Po hardwarové stránce, byl první iPhone unikátní. Inženýrům ze společnosti Apple se podařilo umístit do 115 milimetrů vysokého, 61 milimetrů širokého a 11,6 milimetrů tenkého těla umístit 2 megapixelový fotoaparát, Wi-Fi modul, GSM modul, 3,5 palcový Multi-touch displej a mnoho dalších na svou dobu revolučních komponent [3].

Právě Multi-touch displej přinesl uživatelům prvního iPhonu mnoho výhod v oblasti ovládání chytrého zařízení a iPhone se tak stal velmi populární. Každým rokem představuje Apple vylepšenou verzi iPhonu. Tímto krokem se snaží s každou novou generací přinést nejmodernější technologie a technologické postupy výroby.

Na podzim roku 2014 Apple poprvé představil dva modely iPhone s různou uhlopříčkou. Představený model s názvem iPhone 6 má 4,7 palcový displej a model iPhone 6 plus má 5,5 palcový displej. Jednotlivé modely iPhonu se prodávají v různých barvách a velikostech paměťového úložiště. Od představení prvního iPhonu až do konce roku 2014 se prodalo přibližně 700 miliónu telefonů iPhone [4].

2.2.2 iPod Touch

První generace tohoto zařízení byla představena 7. září 2007. Zjednodušeně lze říci, že se jedná o iPhone bez možnosti telefonování a posílání SMS zpráv. Zařízení obsahovalo všechny důležité prvky jako 3,5 palcový Multi-touch displej, Wi-Fi modul, přehrávač hudby a další. iPod Touch byl, tedy brán jako přehrávač hudby obohacený o výhody chytrých zařízení.

Poslední generace iPodu Touch byla představena 12. září 2012. Tato generace přinesla větší 4 palcový displej, lepší a výkonnější hardware a také zadní poutko, pro možnost uchycení iPodu na ruce. Jednotlivé modely iPod Touch se při prodeji liší velikostí paměťového úložiště a barvou [5].

2.2.3 iPad

Stejně jako u představení prvního iPhonu, tak i při představení první generace iPadu šlo o revoluci na trhu s mobilními zařízeními. První iPad představil Steve Jobs 27. ledna 2010 na tiskové konferenci Applu v San Francisku. iPad patří do rodiny chytrých zařízení od Applu a obsahuje tedy podobné funkce jako iPhone, ale opět bez možnosti telefonování či psaní SMS zpráv.

Hlavním poznávacím prvkem iPadu je veliký displej. První model měl 9,7 palcový displej. Díky velkému displeji se iPad hodí ke konzumaci obsahu, jako jsou knihy, webové stránky či fotky. iPad se také velmi dobře uchytil na školách, kde slouží jako doplněk výuky. 23. října 2012 Apple představil zmenšenou verzi iPadu, kterou pojmenoval přídomkem iPad mini. iPad mini díky zmenšení své velikosti také prodělal zmenšení velikosti displeje a to na 7,9 palce. Poslední generace iPadu a iPadu mini představených na podzim 2014, obsahuje nové technologie, mezi které patří snímač otisků prstů Touch ID a výkonnější procesor [6].

3 Dostupné geolokační hry v App Store

3.1 Ingress

Tato hra byla vydána společností Google 19. prosince 2013. Zpočátku byla tato hra pouze pro platformu Android. Verze pro iOS byla vydána až 14. července 2014. Hra využívá prvky Augmented reality. Hra je založena na určování pozic hráče pomocí GPS a také využívá informace o významných místech ve světě. Ve hře proti sobě bojují dvě frakce, Osvícení a Odpor. Cílem hráčů je hledání fiktivních bodů nabitých energií. Tyto energetické body se nacházejí v portálech. Jednotlivé portály jsou umístěny na místech, kde se vyskytují umělecká díla, významné stavby či jiné památky [7].

3.2 Geocaching

Geocaching je název pro hru, jejíž smyslem je hledání krabiček umístěných různě po světě. V těchto krabičkách jsou uloženy různé předměty. Hráči poté pomocí mapy a GPS souřadnic hledají tyto krabičky. Při nalezení krabičky, může hráč uložený předmět vzít a nahradit vlastním předmětem. S rostoucím zájmem o chytré telefony, přišla firma Groundspeak s vlastní aplikací Geocaching určenou pro mobilní zařízení. Aplikace zobrazuje pozice krabiček a hráče na mapě. Díky tomuto zobrazení může tuto hru hrát i běžný uživatel bez potřeby vlastnit dedikované GPS zařízení.

V App Store jsou dostupné i jiné alternativy pro hraní hry Geocaching. Mezi ně patří například aplikace Geosphere, Geocache Viewer anebo Geocaching Buddy [8].

3.3 Traveler's quest

Geolokační hra Traveler's quest byla vydána studiem Kitty Code. Základem této hry je hledání virtuálních pokladů. Pro nalezení pokladu je potřeba zakoupit mapu. Na této mapě jsou pak zobrazena jednotlivá místa s poklady. Pro získání pokladu z mapy, je potřeba se označenému místu co nejvíce přiblížit. Čím blíž je hráč místu s pokladem, tím více je odměněn. Hráč si poté může rozmyslet, zda si daný poklad ponechá, či jej vymění za peníze. Ve hře lze také nalezené poklady opět ukrýt. Čím déle je ukrytý, tím více nabývá na hodnotě. Hráč má také možnost nakoupit vylepšení či pasti s falešnými poklady. Do hry je též integrován žebříček hráčů s nejlepšími výsledky [9].

3.4 CodeRunner

CodeRunner je počinem vývojářského studia RobotChicken Interactive. Hra je zasazena do špionážního prostředí a snímá pozice hráčů. Podle jejich pozice poté vybírá mise, které musí hráči splnit. V misích má hráč za úkol například nabourání se do sítě, získat video záznam z bezpečnostní kamery anebo odposlouchávat hovory. Všechny tyto úkoly jsou pouze fiktivní a jsou zobrazeny na mapě. Hráči mohou také nové mise vytvářet a umisťovat je na mapě podle své současné pozice. Ve hře je obsažen velmi kvalitní dabing, který umocňuje herní zážitek [10].

3.5 Swarm by Foursquare

Aplikace Swarm vznikla rozdělením aplikace Foursquare. Rozdělením došlo k přesunutí původních sociálních mechanismů do aplikace Swarm a aplikace Foursquare si ponechala pouze vyhledávání restaurací, hotelů, kaváren a jiných volnočasových míst. Důvodem společnosti Foursquare Labs pro vytvoření separátních aplikací, bylo to, že původní aplikace Foursquare negenerovala společnosti žádný zisk. Samotná aplikace Swarm by Foursquare si tedy ponechala pouze funkce pro sdružování přátel, plánování aktivit, posílání zpráv a zobrazování přátel v uživatelově okolí. Aplikace integruje rozšíření pro sociální sítě Facebook a Twitter [11].

3.6 Munzee – Scavenger Hunt

Munzee – Scavenger Hunt je hra vytvořená společností Munzee. Hra má velmi podobný herní princip jako Geocaching. Rozdílem je, že ve hře Munzee musíte při nalezení krabičky či místa naskenovat QR kód pomocí fotoaparátu či NFC. Oskenováním kódu potvrzujete, že jste krabičku či místo opravdu nalezli. Tímto způsobem je velmi dobře ošetřeno podvádění ve hře. Za nasbírané kódy získáváte body, ze kterých se vypočítává hráčovo umístění v celosvětovém žebříčku. Ve hře je zobrazena mapa, na které jsou umístěny pozice jednotlivých míst či krabiček v okolí hráče. Tímto je hráči usnadněno hledání a sbírání kódů [12].

4 Použité technologie

4.1 Objective–C

Objective-C je objektově orientovaný programovací jazyk. Tento jazyke je založen na procedurálním programovacím jazyce C s nadstavbou objektově orientovaného jazyka SmallTalk-80. Právě funkcionalita z jazyka SmallTalk-80 dala Objective-C vlastnosti objektově orientovaného jazyka. Programovací jazyk vytvořili Ph.D. Brad Cox a Dr. Tom Love na začátku 80. let v laboratořích Bellu.

Hlavním důvodem pro vytvoření nového jazyka, bylo to, že objektově orientovaný programovací jazyk SmallTalk, byl velice pomalý. Z těchto důvodů zkombinovali rychlost jazyka C a jazyka SmallTalk, a vytvořili Objective-C. Rané verze Objective-C byli jen preprocesorem, který převáděl Objective-C kód do čistého C. V této verzi chyběly knihovny a jazyk pouze obsahoval třídu Object. Ta se starala o alokaci a dealokaci objektů, a o volání jejich metod.

V roce 1988 si Objective-C licencovala společnost NeXT, kterou založil Steve Jobs poté, co byl nucen odejít z Apple Computers. Společnost NeXT vyvinula vývojové nástroje postavené právě na Objective-C. Dalším důležitým rokem pro Objective-C byl rok 1994. V té době vydala firma Sun vývojové prostředí OpenStep založené na Objective-C. OpenStep přinesl do Objective-C nové knihovny, které se staraly o čítače referencí, správu paměti a další. Sun přestal OpenStep dále rozvíjet, protože začal pracovat na svém vlastním programovacím jazyce Java. Do týmu, který pracoval na Javě, Sun přesunul i část lidí z týmu, který pracoval právě na Objective-C. Díky tomu dnes nalezneme v Javě jistou podobnost s Objective-C.

O další rozvoj se postarala až firma Apple, která koupila firmu NeXT a s tím i veškeré vývojové nástroje pro Objective-C. Z vývojových nástrojů NeXTu vzniklo současné vývojové prostředí Xcode[1] a z rozšiřujících knihoven vznikla knihovna pojmenovaná Cocoa. Poslední změnou prošel jazyk Objective-C s příchodem verze 2.0. V této verzi přibyli nové funkcionality jako property, garbage collector a bloky [13].

```
+ (void)persistPlayer:(Player *)player {
    NSUserDefaults *userDefaults = [NSUserDefaults standardUserDefaults];
    NSData *data = [NSKeyedArchiver archivedDataWithRootObject:player];
    [userDefaults setObject:data forKey:@"player"];
    [userDefaults synchronize];
}
```

Výpis 1: Ukázka zdrojového kódu jazyka Objective-C

4.2 Java

Java patří mezi objektově orientované jazyky. Java vznikla na začátku 90. let ve společnosti Sun pod vedením Ph.D. Jamese Goslinga. Nejdříve se vyvíjený jazyk jmenoval Oak. Poté byli inženýři nuceni Oak přejmenovat na dnes známý název Java a to z důvodu, že programovací jazyk s názvem Oak již existoval.

[1] Webová stránka s informacemi o vývojovém prostředí Xcode https://developer.apple.com/xcode/

Java se jako jedna z prvních programovacích jazyků, kompilovala do bajt kódu. Výsledek kompilace se poté spouštěl v Java Virtual Machine. V tom byla veliká výhoda oproti ostatním jazykům v té době, protože Java mohla běžet na různých platformách a architekturách. Syntaxe jazyka Java vychází z jazyků C a C++. Java také implementuje garbage collector, který se stará o správu paměti za programátora. Díky těmto výhodám se Java stala velmi oblíbeným jazykem a rychle se rozšířila mezi vývojáře.

V současnosti je Java licencována společností Oracle, protože v roce 2010 došlo k odkoupení společnosti Sun právě společností Oracle. Implementace Javy je společnosti Oracle rozdělena na dva balíky. První z nich je Java Runtime Enviroment. Balík obsahuje vše potřebné pro spuštění programu zkompilovaném v bajt kódu. Druhý balík se jmenuje Java Development Kit a ten je určen pro vývojáře. V balíku jsou obsaženy nástroje pro kompilaci, dokumentaci, debuggování a jiné. Poslední verzí Javy je verze s číslem 8. Tato verze implementuje lambda funkce, nové knihovny pro práci s časem, paralelní pole a další vylepšení [14].

```
class Pozdrav {
    private String pozdrav;
    public pozdrav(String pozdrav) {
        this.pozdrav = pozdrav;
        System.out.println(this.pozdrav);
    }
}
```

Výpis 2: Ukázka zdrojového kódu jazyka Java

4.3 SQL

Jazyk Sequel vznikl v laboratořích firmy IBM v roce 1974 pod vedením Donalda Chamberlina a Raymonda Boyce. Jeho první použití se objevilo v databázovém systému System R od společnosti IBM. Jazyk měl programátorům ulehčit získávání dat z databáze za pomoci jednoduché syntaxe, která vycházela z anglického jazyka.

Jazyk se rychle rozvíjel, až došlo v roce 1986 k jeho standardizaci organizací ANSI. V roce 1987 byl také standardizován organizací ISO. Přijetím standardů se jazyk Sequel přejmenoval na SQL. První verze standardizovaného jazyka SQL z roku 1986 nese označení SQL-86 právě podle roku přijetí.

Důležitým rokem pro SQL byl rok 1989, kdy byla vydána nová verze jazyka, která podporovala omezení integrity dat. Poslední verzí jazyka SQL je verze z roku 2011. Tato verze přináší podporu pro řetězení DML, vylepšení práce s kolekcemi a jiné.

Díky rychlosti a jednoduchosti syntaxe najdeme podporu SQL ve většině současných relačních databázích a systémech [15].

```
SELECT * FROM player WHERE online = 1 AND age < 18;
UPDATE player SET first_name = 'Adam' WHERE id = 12;
```

Výpis 3: Ukázka zdrojového kódu jazyka SQL

4.4 REST

Poprvé se pojem REST objevil v disertační práci Roye Fieldinga, který se dříve podílel na vzniku HTTP protokolu. Díky jeho bývalé práci, je REST velice úzce spojen právě s HTTP protokolem. REST definuje rozhraní pro distribuované prostředí orientované na data. Nejedná se tedy o přímé volání procedur jako je tomu například u technologií RPC či SOAP. REST tedy usnadňuje přístup k datům a stavu vaší aplikace.

Nejčastěji se REST používá právě s HTTP protokolem, ale lze použít i jiný přenosový protokol. Smyslem technologie REST je posílání požadavků do různých zdrojů, v praxi to znamená, že se nepoužívá pro přístup pouze jeden centrální zdroj. U HTTP protokolu se používají pro přístup ke zdrojům metody popsané v níže uvedené tabulce.

Název metody	**Použití metody**
GET	Získání a čtení dat ze zdroje
PUT	Nahrazení existujících dat ze zdroje
POST	Vytváření nových dat
DELETE	Mazání existujících dat ze zdroje
PATCH	Úpravy existujících dat ze zdroje

Tabulka 3: Přehled metod HTTP protokolu

Další výhoda použití právě HTTP protokolu pro REST je, že HTTP protokol implementuje stavové kódy. Ty jsou užitečné pro hlášení nejrůznějších chyb. Mezi nejčastější stavové kódy patří kód 200, který slouží pro oznámení, že vše proběhlo v pořádku anebo 404, který značí, že daný zdroj nebyl nalezen.

Veliký důraz by se také měl klást na bezstavovost rozhraní. To znamená, že každý požadavek na zdroj by měl být nezávislý na ostatních požadavcích. Z toho vyplývá, že pro autentifikaci uživatele a následného uložení by neměli být použity technologie, jako jsou sessions nebo cookies. To znamená, že každý požadavek by měl obsahovat autentifikační údaje uživatele [16].

4.5 JSON

JSON je formát pro datový zápis objektů vycházející z programovacího jazyka JavaScript. Tento formát vytvořil Douglas Crockford. Hlavním důvodem vzniku formátu JSON byla složitost formátu XML. JSON na rozdíl od XML má jednodušší notaci a programátor nemusí pracovat s DOM. Z těchto důvodů se stal JSON velmi oblíbený při tvorbě webových stránek nebo při komunikaci přes REST rozhraní. Do formátu JSON můžeme uložit data typu řetězec, číslo, logická hodnota, prázdná hodnota, pole či opět objekt. Ostatní datové typy musíme před uložením do JSONu serializovat a poté při čtení zpět deserializovat [17].

Díky jednoduchosti a velké popularitě je pro formát JSON vyvinuta řada knihoven pro většinu programovacích jazyků. Seznam knihoven pro jednotlivé programovací jazyky je uveden na oficiálních stránkách o formátu JSON[2].

```
[
    {
        "player": {
            "id": 10,
            "first_name": "Adam",
            "last_name": "Zikmund",
            "age": 21,
            "online": true,
            "position": {
                "latitude": 28.487122,
                "longitude": 48.457421
            }
        }
    }
]
```

Výpis 4: Reprezentace objektu hráče v JSON formátu

[2]Seznam knihoven pro jednotlivé programovací jazyky je na webových stránkách http://www.json.org

5 Herní mechanismus

5.1 Základní herní princip

Hlavním cílem práce je vytvoření hry pro platformu iOS, která vychází z pravidel deskové hry Fantom staré Prahy. V této hře se z hráčů vybírá jeden fantom a zbylí hráči mají za úkol jej dopadnout. Fantom se snaží hráčům vyhýbat, tak aby nebyl dopaden. Hra končí v případě, že je fantom dopaden nebo v případě, že se všem hráčům úspěšně vyhýbá. Tento herní princip bylo potřeba přesunout do mobilních zařízení s využitím dostupných technologií, které se dnes nabízejí. A tak byla herní deska nahrazena za reálnou mapu světa a z figurek se stali opravdoví hráči, kteří svůj pohyb na mapě zaznamenávají pomocí GPS.

5.2 Příprava hry

Vytvořit novou hru může jakýkoliv hráč, který již není hráčem v probíhající hře či není vlastníkem již vytvořené nové hry. Hráč při vytváření nové hry může nastavit různé vlastnosti, které ovlivní její průběh. Hráč, který vytváří novou hru, má poté oprávnění ji spustit, zrušit či vyloučit některého z připojených hráčů.

5.3 Započetí hry

Po připojení potřebného počtu hráčů, může zakládající hráč hru spustit. Pro spuštění hry musí být připojeni do hry alespoň dva hráči. Z toho důvodu, aby byl ve hře vždy alespoň jeden fantom a jeden běžný hráč, který se snaží fantoma dopadnout. Při spuštění hry se ze všech připojených hráčů vybere jeden fantom. Do již spuštěné hry nebudou mít hráči možnost se připojit.

5.4 Zapojení se do vytvořené hry

Hráč, který nechce zakládat vlastní hru, se může připojit do již existující hry. Hráč bude mít možnost vybrat si hru ze seznamu, do které se chce připojit. Jednotlivé hry bude možno třídit podle předem stanovených kritérií. Po připojení do hry, musí hráč čekat do doby, než ji zakládající hráč spustí. Pokud hráč nechce již setrvat ve hře, má oprávnění ji opustit a to pouze v případě, že hra nebyla již spuštěna.

5.5 Průběh a ukončení hry

Po spuštění začíná hra podle předem zvolených nastavení. Fantom ze začátku podléhá ochraně před dopadením. A to hlavně proto, aby měl fantom čas na ukrytí před ostatními hráči. Po uplynutí této doby, mohou hráči dopadnout fantoma. Všichni hráči ve hře budou vyobrazeni na mapě na základě jejich současné polohy. Díky tomu mohou hráči plánovat svůj budoucí pohyb tak, aby dopadli fantoma. Toho může také využít fantom a úspěšně se vyhýbat ostatním hráčům.

K samotnému ukončení probíhající hry dochází v těchto třech případech:

- Fantom byl dopaden hráčem, který se k němu přiblížil na určitou vzdálenost
- Ukončením hry zakládajícím hráčem
- Fantom nebyl dopaden v časovém limitu
- Fantom byl neaktivní

V prvním případě dochází k ukončení hry, když se jeden z hráčů přiblíží fantomovi natolik, až může být fantom dopaden. V druhém případě nastává ukončení hry tehdy, když hráč zakládající hru ukončí. Třetí možností je, když se fantom úspěšně vyhýbá ostatním hráčům po určitou dobu. Poslední možností je, pokud je fantom delší dobu neaktivní, to znamená, že doba jeho poslední změny polohy přesáhla určitý limit.

5.6 Obtížnost hry

Průběh hry je také ovlivněn nastavením obtížnosti hry. První, lehčí, obtížnost dává větší výhodu chycení fantoma hráčům. A to z důvodů, že fantomova pozice na mapě souhlasí s jeho současnou pozicí. V případě těžší obtížnosti je, zobrazení pozice fantoma na mapě opožděna a nesouhlasí tak s jeho aktuální pozicí. To dává výhodu fantomovi v tom, že může svůj pohyb předem promyslet.

Komplexnost hry bude možné ovlivnit dalšími nastaveními hry. První možností je určit limit pro maximální počet hráčů. V případě naplnění tohoto limitu bude znemožněno připojení ostatních hráčů do hry. Druhou možností je nastavení názvu hry. Název hry je důležitým poznávacím prvkem při vyhledávání hry ostatními hráči. Poslední možností je nastavení doby ochrany fantoma. Takto nastavená doba poté chrání fantoma před dopadením ostatními hráči.

6 Analýza a návrh řešení

Před výběrem technologií a samotným vývojem hry je potřeba provést analýzu. Podle výsledné analýzy se následně zvolí optimální návrh řešení a technologie k němu potřebné. Analýza je rozdělena na dva logické celky: analýza pro serverovou část a analýza pro klientskou část. V serverové časti je důraz směřován na analýzu perzistentní vrstvy, komunikačního rozhraní a herní logiky. V klientské části je pak kladen důraz na analýzu uživatelského rozhraní a způsobu určování polohy hráče.

6.1 Server

Při realizaci hry je potřeba vytvořit centrální server, který se bude starat o všechny požadavky klientů. Mezi tyto požadavky patří například: registrace nového hráče, přihlášení hráče, připojení hráče do hry a jiné. Tyto požadavky jsou vynuceny přímo klientem a server je bude zpracovávat nezávisle na sobě. Server bude také nezávisle provádět procesy na pozadí, které se starají o správný průběh hry.

Další důležitou funkcí serveru je schopnost ukládání dat. Tyto data je potřeba uchovávat trvale a v případě potřeby je prezentovat klientům v závislosti na jejich požadavcích.

6.1.1 Perzistentní vrstva

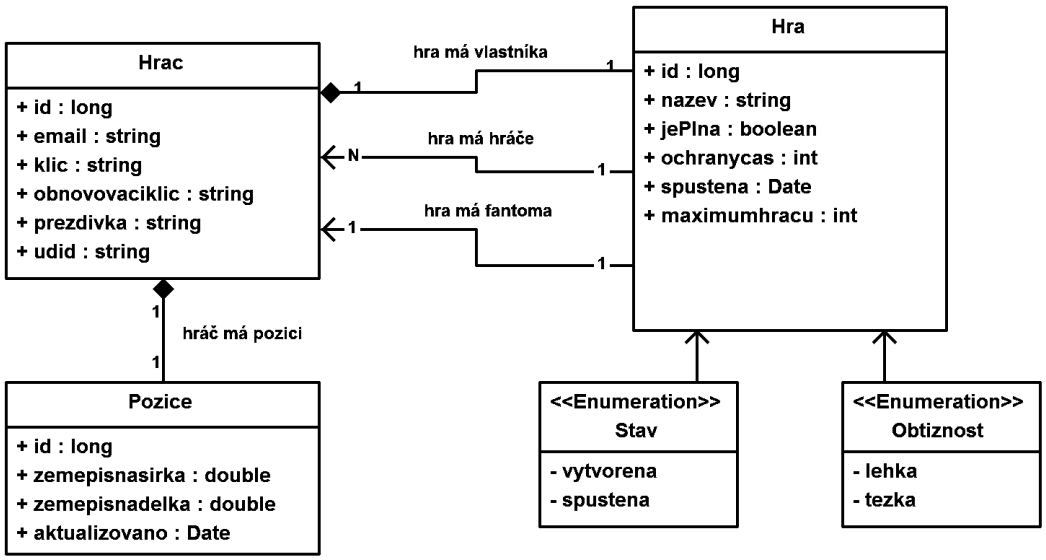
V analýze perzistentní vrstvy je potřeba brát zřetel na data, která budou uložena a také na jejich množství. V případě hry bude potřeba uchovávat data a informace o:

- Hráčích
- Vytvořených hrách
- Současných pozicích hráčů

U jednotlivých hráčů je potřeba uchovávat informace o jejich emailu, přezdívce, heslu, kódu pro resetování zapomenutého hesla, UDID zařízení a autentifikačním tokenu. Pro vytvořené hry je nutné uchovat informace o názvu, stavu, maximálním počtu hráčů, obtížnosti, době ochrany fantoma, času spuštění, a zdali je hra plná. V případě ukládání současné polohy hráče se uchovává zeměpisná šířka, zeměpisná délka a čas kdy byla pozice aktualizována.

Z předem definovaných pravidel hry pak vyplývají jednotlivé vztahy mezi těmito entitami. To znamená, že hráč uchovává informaci o aktuální hře, ve které je připojen a také informaci o své současné pozici. Ve vytvořené hře je pak potřeba si uchovat informaci o tom kdo z hráčů je fantom a kdo vlastník.

Díky rozšířenosti a velké popularitě relačních databázových systémů je právě tato technologie zvolena k řešení perzistentní vrstvy. K usnadnění implementace řešení pro relačně databázový systém je použit třídní diagram.



Obrázek 1: Třídní diagram perzistentní vrstvy

6.1.2 Komunikační rozhraní

Návrh rozhraní pro zpracování požadavků klienta je velmi důležitou součástí serverové části. Jelikož server musí poskytovat odpovědi na požadavky všech klientů. Mezi požadavky, které musí rozhraní zpracovávat, patří například:

- Registrace hráče
- Přihlášení hráče
- Připojení hráče do hry
- Vyloučení hráče ze hry
- Spuštění hry

Všechny tyto požadavky je potřeba důkladně zanalyzovat. Každý z požadavků může klást jiné nároky na server a také může pracovat s jinými typy dat. Proto je potřeba u každého požadavku klienta, zjistit s jakými daty operuje, jakou očekává odpověď a jaké případy mohou nastat při zpracovávání daného požadavku.

Také je potřeba brát zřetel na bezpečnost rozhraní, jelikož komunikace mezi serverem a klientem bude zprostředkována skrz síť internetu. V takovémto případě je potřeba veškerá citlivá data, u kterých hrozí možnost zneužití útočníkem, řádně zabezpečit. V případě, že by tato data nebyla zabezpečena, může dojít k tomu, že útočník data zneužije ve svůj prospěch anebo se pokusí získaná data využít k napadení systému.

Důležitým aspektem při analýze rozhraní je také snaha o co nejmenší datové nároky pro přenos odpovědi ke klientovi. Kdyby se při analýze a návrhu nekladl důraz na tento

aspekt, mohlo by v brzké době dojít k tomu, že propustnost datové linky nebude dostatečně velká pro transport potřebného objemu dat a dojde tak k zahlcení celého systému.

S přihlédnutím ke všem těmto bodům analýzy je zvolen pro řešení komunikačního rozhraní standart REST. Tento standart je popsán výše a díky tomu, že je úzce propojen s protokolem HTTP, tak i při návrhu řešení komunikace je použit právě tento protokol. Protokol HTTP umožňuje zpracovávat jednotlivé požadavky klientů nezávisle na sobě. Právě nezávislé zpracovávání jednotlivých požadavků klienta je jedním ze stěžejních bodů standartu REST.

Všechny předchozí kroky vedly k analýze a návrhu řešení pro komunikaci se serverem na vyžádání klienta. Ale v případě opačného směru komunikace, tedy takového kdy server posílá požadavek klientovi, je potřeba zvolit jiné řešení.

Analýza rozhraní pro komunikaci kdy server posílá požadavek klientovi je odlišná. Jelikož tato komunikace je v případě hry pouze informativní a požadavky serveru na klienta pouze přenášejí informace o akcích, které proběhly na serveru. Jednotlivé požadavky mají notifikační charakter a obsahují jen krátkou informaci o akci, která proběhla. Mezi notifikace, které jsou odeslány na základě provedených akcí na serveru, patří například:

- Hráč byl vyloučen ze hry
- Hra byla zrušena
- Hráč se připojil do hry
- Ukončení hry z důvodu neaktivity fantoma

Návrh řešení pro tuto problematiku je jednoznačný. Důvodem je právě Apple, protože podporuje pouze vlastní službu pro notifikace. Proto v návrhu řešení a následné implementaci musí být zahrnuta tato technologie.

6.1.3 Herní logika

Většina herní logiky se odehrává právě na serveru. To přináší výhodu lepší kontroly nad akcemi jednotlivých hráčů a omezuje se tak zneužití herních mechanismů. Toto řešení přináší také nevýhodu v tom, že většina logických operací vyhodnocuje server. To znamená, že většina výpočetní zátěže je směřována právě na server a není tak možnost zpracovávat tuto zátěž na klientských zařízeních.

Analýza herní logiky je značně ovlivněna pravidly hry. Jelikož jednotlivé funkce, které se starají o průběh hry, musí vycházet z předem stanovených pravidel. Do těchto funkcí je potřeba také zahrnout jistou dynamičnost, jelikož pravidla hry se mohou měnit podle toho, jaké nastavení vlastník hry zvolil při vytváření nové hry. V analýze herní logiky by měl být také kladen důraz na optimalizaci náročnosti jednotlivých logických operací, které budou funkce provádět. Příklady akcí, které je potřeba vyřešit ve funkcích, jsou například:

- Ověření zda je fantom aktivní

- Výpočet vzdálenosti mezi fantomem a hráčem
- Ověření zda je fantom stále chráněn časovým limitem

Řešením, takových operací je vytvoření algoritmu, který bude jednotlivé akce zpracovávat a vracet požadovaný výsledek. Na základě výstupu z takového algoritmu mohou být prováděny další akce, které ovlivňují herní průběh.

6.2 Klient

Při analýze a návrhu řešení klientské části je potřeba brát v potaz, že cílem práce je realizace hry pro platformu iOS. Proto je nutné veškeré kroky přizpůsobit právě možnostem této platformy. A to hlavně z toho důvodu, že Apple kontroluje chování a bezpečnost aplikací při schvalovacím procesu do App Store. K těmto restrikcím patří i doporučení Applu, jak aplikaci navrhovat a implementovat. Soubor těchto doporučení se jmenuje Human Interface Guidelines[3].

Hlavní úloha klienta, je prezentace dat uživateli v srozumitelné podobě a také možnost provádění akcí navazujících na komunikační rozhraní serveru. Mezi další funkční úlohy klienta patří vyobrazení hráčů a fantoma na mapě, určování současné polohy hráče a také zobrazování notifikačních zpráv.

6.2.1 Uživatelské rozhraní

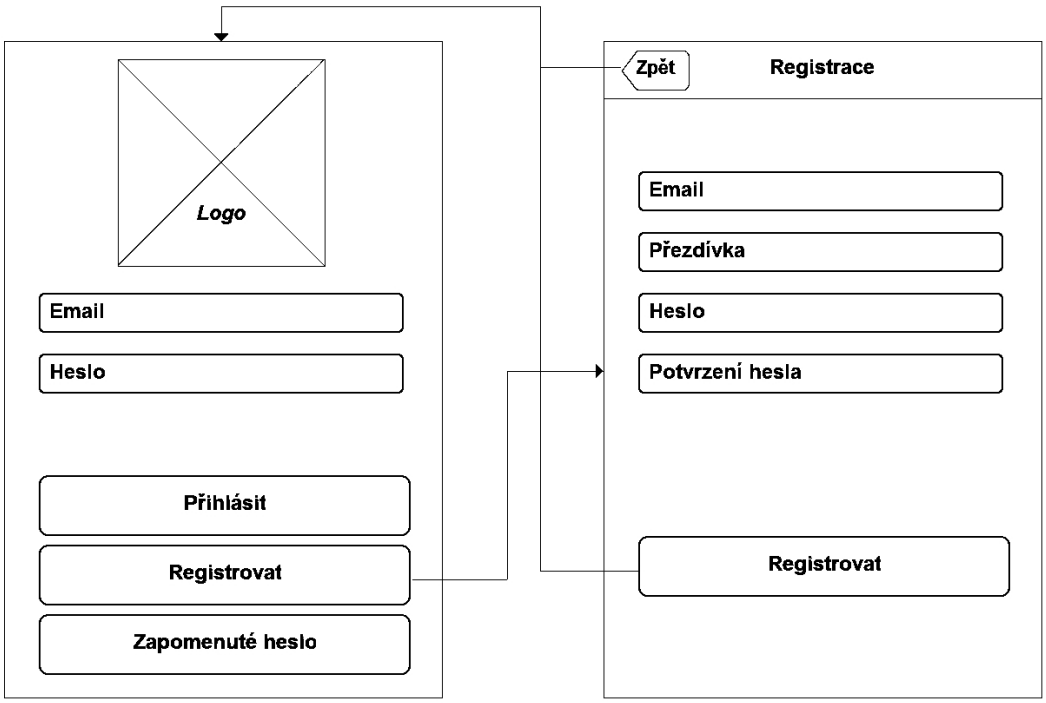
U návrhu uživatelského rozhraní aplikace je nutno dbát na výslednou přehlednost a snadnou orientaci pro uživatele v aplikaci. Proto vše co je uživateli zobrazeno musí mít logické uspořádání na obrazovce a nesmí nastat situace, kdy uživatel neví, co daný prvek provádí či zobrazuje. Důraz by měl být také kladen na jednoduchost uživatelského rozhraní. Jednoduchostí návrhu by měl být dosažen výsledek, kdy uživateli nebude zobrazeno příliš mnoho dat najednou. V opačném případě by opět docházelo ke zmatení a dezorientaci uživatele v aplikaci. Do návrhu by také měla být zahrnuta logická posloupnost a snadná navigace mezi jednotlivými obrazovkami, které zobrazují data a grafické prvky.

Dalším krokem při návrhu je způsob, jakým bude aplikace lokalizována do různých jazyků. V případě opomenutí tohoto kroku by mohlo dojít k jisté ztrátě srozumitelnosti pro uživatele z odlišně mluvících zemí.

Výsledným řešením by měly být grafické podklady ve vysoké kvalitě a návrh jednotlivých obrazovek s přesným umístěním grafických prvků. K výslednému řešení napomáhají právě výše uvedená doporučení Applu.

Následující obrázek je návrhem obrazovky pro přihlášení a registraci hráče. Návrh obsahuje navigační prvky, rozmístění tlačítek a textových polí.

[3] Uvedený soubor doporučení se nachází na webové stránce
https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/MobileHIG/



Obrázek 2: Návrh uživatelského rozhraní přihlašovací a registrační obrazovky

6.2.2 Určování polohy

Určování současné polohy je stěžejní funkcionalita pro průběh hry. Proto při analýze této části je potřeba zjistit, s jakou přesností je současná poloha určována. Dále zda je poloha určována i v případě, že aplikace není na klientském zařízení spuštěna. A jak často dochází k určování polohy. Všechny tyto tři části jsou důležitým krokem analýzy k následnému vyhodnocení řešení.

Díky tomu, že platforma iOS podporuje službu GPS pro získávání polohy, je k řešení použita systémová knihovna z iOS, která veškerou funkcionalitu obstarává. Tato knihovna poskytuje veškeré informace o pozici uživatele, přesnosti naměřené pozice a okolnosti za jakých má být současná pozice určována.

7 Implementace

K vývoji byly použity diagramy a všechny informace získané z analýzy a návrhu. Při implementaci bylo potřeba vytvořit každou část samostatně. Jako první bylo potřeba naimplementovat serverovou část. U serveru bylo nutné vytvořit perzistentní vrstvu a aplikační vrstvu. Po dokončení serverové části přišel na řadu klient. Při vývoji klienta byl kladen důraz na dodržení MVC architektury. V klientské části bylo potřeba naimplementovat uživatelské rozhraní, komunikaci se serverem a část pro určování současné polohy hráče.

7.1 Server

Implementace serveru je rozdělena na dva samostatné projekty. První projekt PhantomModel obsahuje pouze perzistentní vrstvu a vše potřebné pro její zprovoznění. Druhý projekt PhantomAPI obsahuje referenci na první projekt a také aplikační vrstvu rozdělenou do následujících balíčků:

- Api – v balíčku jsou třídy pro chod komunikačního rozhraní
- DTO – balíček obsahuje implementaci přepravek, které se využívají pro přenos dat po síti
- Exception – balíček obsahuje jednotlivé výjimky
- Helper – v balíčku jsou pomocné třídy, které zajišťují odesílání elektronické pošty či vytváří pomocnou datovou strukturu pro odpovědi
- Service – balíček obsahuje třídy, které zajištují logickou část aplikace

7.1.1 Perzistentní vrstva

Při implementaci perzistentní vrstvy byla použita technologie JPA, která je součástí Java Enterprise Edition. Tato technologie usnadňuje programování objektově relačního mapování za použití anotací. Anotace ovlivňují chování třídních proměnných, mapování a také umožnují tvorbu parametrizovaných dotazů. Následující tabulka znázorňuje seznam nejpoužívanějších anotací při implementaci:

Anotace	Popis
@Id	Vytváří primární klíč
@GeneratedValue	Automaticky generuje hodnotu
@Column	Nastavuje vlastnosti atributu
@OneToOne	Automaticky vytváří vazbu 1 : 1 mezi třídami
@NamedQuery	Vytváří parametrizovaný dotaz za použití SQL

Tabulka 4: Přehled použitých JPA anotací

Jednotlivé třídy byly vytvořeny na základě třídního diagramu. V každé třídě byly vytvořené privátní proměnné a jejich set a get metody. Nad jednotlivé proměnné byly umístěny anotace podle potřeby. Vazby vyplývající z třídního diagramu byly vytvořeny za pomoci mapovacích anotací. Následně byly naimplementovány parametrizované dotazy pro složitější operace s daty.

7.1.2 Komunikační rozhraní

Při implementaci komunikačního rozhraní bylo potřeba vytvořit třídy, které budou mít za úkol přijímání HTTP požadavků. Jednotlivé funkce v těchto třídách, pak měly přijaté požadavky dále zpracovávat podle jejich adresy, typu požadavku a parametrů. Pro řešení byla použita technologie JAX-RS. Tato technologie je součástí Java Enterprise Edition. JAX-RS usnadňuje implementaci REST API v jazyce Java. U této technologie se používají anotace podobně jako u JPA. Následující tabulka ukazuje seznam použitých anotací při implementaci:

Anotace	Popis
@POST	Definuje, zda daná funkce bude přijímat POST požadavek
@Path	Definuje cestu, na které bude funkce očekávat požadavek
@Produces	Definuje, jaký typ dat bude obsahovat tělo odpovědi
@FormParam	Definuje parametr požadavku, který daná funkce bude přijímat

Tabulka 5: Přehled použitých JAX-RS anotací

Komunikační rozhraní je rozděleno do tří tříd, které mají za cíl přijímat a dále zpracovávat příchozí požadavky. První třída PlayerResource obsahuje funkce potřebné pro obsluhu hráče. Následující tabulka obsahuje seznam a popis zdrojů, které třída PlayerResource obstarává:

Cesta	Parametry	Popis
player/register	email, password, nickname	Vytváří nového hráče
player/resetPassword	email	Resetuje heslo
player/newPassword	email, password, resetcode	Nastavuje nové heslo
player/login	email, password	Přihlásí hráče
player/getInfo	id, token	Získává informace o hráči
player/logout	id, token	Odhlásí hráče
player/addDeviceToken	id, token, deviceToken	Přidá nové UDID hráči

Tabulka 6: Seznam a popis zdrojů třídy PlayerResource

Třída GameResource implementuje funkce potřebné pro ovládání hry. Následující tabulka obsahuje seznam a popis zdrojů, které třída GameResource obstarává:

Cesta	Parametry	Popis
game/create	id, token, name, maximumPlayers, difficulty, catchDelay	Vytváří novou hru
game/search	id, token	Vyhledává všechny vytvořené hry
game/searchByName	id, token, name	Vyhledává všechny vytvořené hry podle jména
game/join	id, token, gameId	Připojí hráče do hry
game/getInfo	id, token	Získává informaci o hře
game/close	id, token	Uzavírá vytvořenou hru
game/start	id, token	Spouští vytvořenou hru
game/leave	id, token	Opouští vytvořenou hru
game/kick	id, token, playerId	Vylučuje hráče z vytvořené hry
game/positions	id, token	Získává pozice hráčů ve hře
game/catchPhantom	id, token	Pokouší se o dopadení fantoma
game/destroy	id, token	Ukončuje spuštěnou hru

Tabulka 7: Seznam a popis zdrojů třídy GameResource

Poslední třída PositionResource zajišťuje aktualizaci současné pozice hráče. Následující tabulka obsahuje popis zdroje, který obstarává aktualizaci současné pozice hráče:

Cesta	Parametry	Popis
position/add	id, token, longitude, latitude	Přidá současnou pozici hráče

Tabulka 8: Seznam a popis zdrojů třídy PositionResource

7.1.3 Herní logika

Herní logika je vyhodnocována podle příchozích požadavků z komunikačního rozhraní. Proto při implementaci funkcí herní logiky je potřeba brát zřetel na parametry, které jednotlivé požadavky obsahují. Herní logika je rozdělná do několika celků, které se starají o vyhodnocování. Tyto celky jsou reprezentovány třídami. Mezi tyto třídy patří:

- PlayerService – implementuje funkce potřebné pro obsluhu požadavků z třídy PlayerResource a také funkci pro ověření správnosti emailové adresy

- GameService – obsahuje funkce, které přijímají požadavky z třídy GameResource a dále také pomocné funkce pro vypočítávání vzdáleností, vybírání fantoma a validaci oprávnění
- PositionService – tato třída má na starosti zpracovávání požadavku ze třídy PositionResource
- AuthService – třída implementuje funkce pro autentifikaci uživatele a generování klíčů
- SchedulerService – funkce této třídy mají za úkol v intervalech kontrolovat, zda je fantom aktivní a zda uplynul limit potřebný pro vítězství fantoma
- ApnQueueService – obsahuje funkce, které odesílají notifikace s informacemi jednotlivým hráčům

Pro funkce v těchto třídách bylo, také potřeba vytvořit speciální výjimky. Hlavním důvodem pro jejich vytvoření je, že každý požadavek musí mít i odpověď. Proto je použití výjimek při neočekávaném průběhu funkce vhodným řešením. Příklad použití výjimky může být při registraci nového hráče. V případě, že uživatel zadá nevalidní email, dojde k vyhození výjimky. Ta bude dále zpracována a odeslána v odpovědi klientovi.

Pokud při provádění funkce nedojde k problému, je potřeba komunikačnímu rozhraní vrátit relevantní odpověď. Pro tuto úlohu bylo potřeba vytvořit generickou třídu Reply. Při vytváření instance této třídy je potřeba do konstruktoru vložit data, která chceme v odpovědi odeslat. Po vytvoření instance této třídy je objekt vrácen návratovou hodnotou do komunikačního rozhraní. Ve funkci z komunikačního rozhraní je následně tento objekt zpracován a odeslán ve formátu JSON klientovi.

7.1.4 Problém při serializaci do formátu JSON

Při zpracovávání dat z objektu třídy Reply došlo k problému se serializací do formátu JSON. Tento problém nastal, když bylo potřeba serializovat objekty třídy Game. Při serializaci docházelo k tomu, že se rekurzivně volaly get metody. Za tímto problémem stály právě cyklické vazby mezi třídami Game a Player. Následně díky těmto cyklickým vazbám docházelo k tomu, že když se funkce dotazovala na data o vlastníkovi hry, došlo k tomu, že při zjišťování dat o vlastníkovi byla opět zavolána funkce, která získávala informace o hře. Tímto způsobem vzniklo cyklické volání požadavků.

Prvním řešením toho problému bylo použití anotací z knihovny Jackson. Tato knihovna se stará právě o serializaci objektů do formátu JSON. Díky anotacím z této knihovny, lze přerušit cyklickou vazbu mezi třídami a vyřešit tak daný problém.

Zároveň s prvním řešením bylo použito i druhé řešení. Druhé řešení spočívá ve vytvoření pomocných DTO tříd. Instance těchto tříd tak zaobalí příslušný objekt, který je potřeba serializovat. Další výhoda tohoto řešení je, že tímto způsobem lze definovat, která data z původního objektu chceme odeslat v odpovědi klientovi. Příkladem může být nežádoucí odeslání hesla uloženého v objektu hráče. Dále tímto řešením dochází ke

zmenšení nároku na datovou propustnost sítě. Jelikož jsou odesílána pouze potřebná data.

7.2 Klient

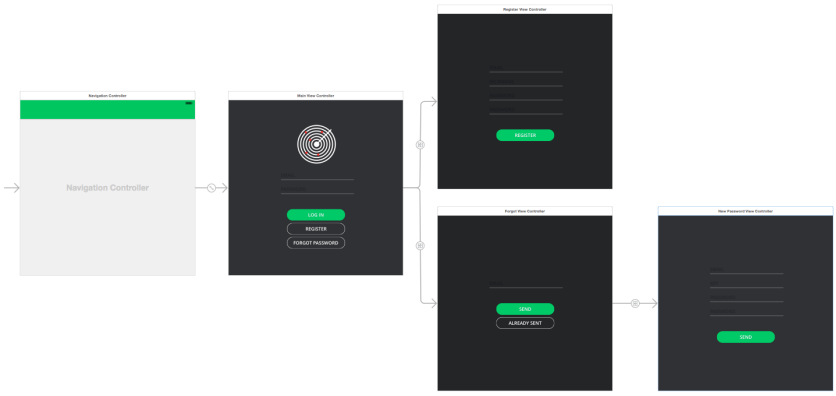
Při implementaci klienta byl kladen důraz na dodržení MVC architektury. Z tohoto důvodu byla řídící logika oddělena od uživatelského rozhraní a datového modelu. Při vývoji těchto části byly použity pouze oficiální iOS knihovny.

7.2.1 Uživatelské rozhraní

Při vývoji uživatelského rozhraní byla použita technologie Storyboard. Tato technologie umožňuje jednoduchý a rychlý návrh uživatelského rozhraní. Mezi hlavní výhody patří:

- celistvý přehled nad jednotlivými obrazovkami
- snadná integrace do zdrojových kódů řídící logiky
- možnost definovat přechody mezi obrazovkami
- lehké umisťování nových prvků uživatelského rozhraní
- jednoduše upravitelné chování a vzhled umístěných prvků
- určitá nezávislost na zbytku aplikace

Uživatelské rozhraní je rozděleno na tři samostatné celky. Prvním z nich je MainStoryboard. Ten obsahuje veškeré obrazovky potřebné pro přihlášení hráče, registraci hráče a resetování hesla. Druhým je SearchStoryboard, který obsahuje obrazovky potřebné pro vyhledání hry, nastavení hry a vytvoření hry. Posledním z nich je GameStoryboard, který obsahuje obrazovku zobrazující průběh hry. Na následujícím obrázku můžeme vidět strom obrazovek z MainStoryboard.



Obrázek 3: Strom obrazovek uživatelského rozhraní z MainStoryboard

7.2.2 Komunikace se serverem

Další důležitou částí je komunikace se serverem. Při implementaci této části bylo potřeba vytvořit funkce, které odesílají požadavky na komunikační rozhraní serveru. Tyto funkce byly volány při určité akci uživatele v aplikaci. Těmto funkcím bylo potřeba předat data, které komunikační rozhraní serveru potřebovalo ke zpracování.

Tyto funkce byly rozděleny do tří tříd. Tyto třídy logicky korespondují s třídami v komunikačním rozhraní serveru. Proto první třída nese název PlayerRequest a její požadavky jsou směrovány na zdroje z třídy PlayerResources. Stejně jako předchozí třída, tak i třídy GameRequest a PositionRequest směřují své požadavky na zdroje do tříd GameResource a PositionResource.

Při implementaci bylo nutné myslet také na to, že při odeslání požadavku je potřeba zpracovat odpověď. Proto bylo potřeba vyvinout funkci pro deserializaci dat ve formátu JSON do objektu jazyka Objective-C. K tomuto řešení posloužila třída NSJSONSerialization, která obsahuje potřebné funkce pro řešení tohoto problému. Dále bylo nutné při vývoji myslet na to, že odpověď může obsahovat výjimku. Tuto výjimku bylo také potřeba zpracovat a v nutném případě zobrazit uživateli chybovou hlášku.

Při odesílání požadavku a zpětném přijmutí odpovědi může dojít k problému s připojením k síti internetu či výpadku serveru. Proto bylo nutné brát při vývoji v potaz i tento problém. V případě, že k takové události došlo, byla uživateli zobrazena chybová hláška.

Poslední problém, který bylo potřeba při implementaci vyřešit je odesílání požadavků a zpracovávaní odpovědi v samostatném vlákně. Důvodem tohoto problému je, že kdyby se všechny tyto procesy zpracovávaly v hlavním vlákně, došlo by k zamrznutí celé aplikace do doby, než by byly dokončeny.

7.2.3 Určování současné polohy

Klíčovou funkcionalitou klientské části je určování současné polohy hráče pomocí GPS. K implementaci této funkcionality byla použita třída CLLocationManager z knihovny CoreLocation. Zmíněná třída zajišťuje zjišťování polohy mobilního zařízení z GPS modulu. Následující úryvek kódu ukazuje potřebnou část pro automatické zjišťování polohy.

```
_locationManager = [[CLLocationManager alloc] init];
[_locationManager setDelegate:self];
[_locationManager requestAlwaysAuthorization];
[_locationManager setDesiredAccuracy:kCLLocationAccuracyBest];
[_locationManager setDistanceFilter:kCLDistanceFilterNone];
[_locationManager setPausesLocationUpdatesAutomatically:NO];
[_locationManager startUpdatingLocation];
```

Výpis 5: Úryvek zdrojového kódu k určování současné polohy

Na prvním řádku dochází k alokaci paměti pro novou instanci třídy CLLocationManager a dále k zavolání výchozího konstruktoru. Výsledná instance se poté přiřadí do třídní proměnné. Na druhém řádku dochází k nastavení delegáta. Tento delegát poté reaguje na příchozí události vyvolané instancí třídy CLLocationManager. Na dalším řádku

dochází vyvolání požadavku uživateli. Po vyvolání musí uživatel souhlasit, že aplikace smí využívat data z GPS modulu. Na čtvrtém řádku dochází k nastavení přesnosti, které chceme při získávání polohy dosáhnout. V našem případě požadujeme nejvyšší možnou přesnost. Další řádek dále nastavuje minimální práh vzdálenosti potřebné pro vyvolání nové události o změně polohy. V tomto případě je nastaven práh na nulu. Šestý řádek říká, že k určování polohy bude docházet i v případě, že aplikace bude přesunuta na pozadí. Posledním řádkem spustíme automatické monitorování polohy.

8 Testování

Před samotným nasazením je potřeba důkladně otestovat různé herní situace. Při vývoji se chování hry testovalo v simulátoru a chování serveru se testovalo na počítači v lokální síti. Po otestování funkčnosti hry na simulátoru a serveru na lokálním počítači je potřeba obě části otestovat, také na reálných zařízeních. Hlavním důvodem pro testování na reálných zařízeních je to, že výsledné aplikace se nemusejí chovat stejně jako v simulátoru.

Při testování hry, bylo potřeba ověřit správnost komunikace se serverem, chování uživatelského rozhraní a také průběh určování současné polohy hráče. U serveru bylo nutno otestovat chování na příchozí požadavky klienta, jednotlivé odpovědi na požadavky, správnost ukládání dat do perzistentní vrstvy a notifikační službu.

Pro testování hry byl použit vývojářský nástroj s názvem TestFlight[4]. Tento vývojářský nástroj poskytuje Apple všem vývojářům, kteří mají předplacený iOS Developer Program[5]. Díky tomuto nástroji je možno testovat aplikace na reálných zařízeních. Výhoda tohoto nástroje je, že aplikace mají stejné chování jako aplikace nasazené přímo do App Store. Další nespornou výhodou je možnost přizvat do testování aplikace i ostatní uživatele s platným Apple účtem. Všechny tyto výhody dávají vývojářům lepší možnost otestovat své aplikace ještě před samotným nasazením do App Store.

Samotnou funkčnost serveru bylo potřeba otestovat také v rámci sítě internetu. Z důvodu, že po nasazení aplikace do App Store bude komunikace mezi klienty a serverem zprostředkována právě skrz síť internetu. Proto bylo potřeba zvolit hosting, který splňoval všechny požadavky pro testování serverové části. Po vybrání vhodného hostingu bylo potřeba serverovou část nasadit a zprovoznit.

8.1 Zkušební hra

Po ověření správnosti všech funkcionalit klientské a serverové části bylo potřeba otestovat samotný průběh hry. Pro otestování průběhu hry v reálném prostředí byla použita zařízení iPad mini 2 a iPad Air 2. Na těchto zařízeních došlo k nainstalování hry pomocí služby TestFlight a následnému spuštění. Po spuštění hry bylo potřeba vytvořit testovací uživatelské účty. Když došlo k úspěšném přihlášení mohlo začít testování hry.

Před samotným testováním probíhající hry, bylo potřeba také ověřit všechny potřebné úkony před započetím hry. Mezi tyto úkony patří:

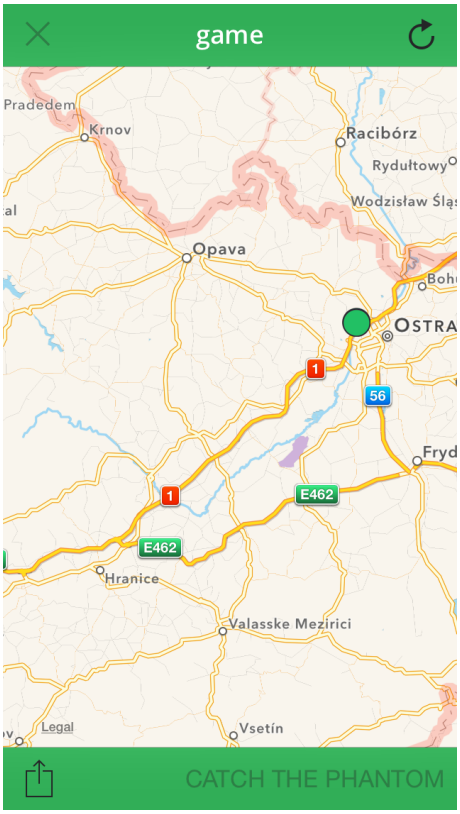
- Vytvoření nové hry hráčem
- Vyhledání a připojení se do hry
- Vyloučení hráče ze hry
- Zrušení vytvořené hry vlastníkem
- Spuštění hry vlastníkem hry

[4]Webová stránka s informacemi o nástroji TestFlight https://developer.apple.com/testflight/
[5]Webová stránka s informacemi o iOS Developer Programu https://developer.apple.com/programs/ios/

Po vytvoření a spuštění hry došlo k ověřování všech funkcionalit, které byly potřeba pro správný průběh hry. Případy, které jednotlivé funkcionality řešily, jsou následující:

- Aktualizace současné polohy všech hráčů a fantoma
- Ukončení probíhající hry vlastníkem
- Ukončení hry díky neaktivitě fantoma
- Ukončení hry po úspěšném dopadení fantoma
- Ukončení hry po úspěšném úniku fantoma před ostatními hráči

Všechny funkcionality nutné pro správný průběh byly během zkušebních her úspěšně ověřeny. Zkušební hry tak otestovaly chování hry v reálném prostředí na reálných zařízeních. Po všech těchto krocích je možné finální klientskou a serverovou část nasadit do reálného provozu.



Obrázek 4: Snímek obrazovky zobrazující průběh testovací hry

9 Nasazení

Po důkladném otestování klientské i serverové časti je možno uvést obě tyto části do reálného provozu. U serveru je nutné zprovoznit prostředí, které umožňuje spuštění a provoz Java Enterprise Edition. Po zprovoznění potřebného prostředí lze nasadit výsledný balíček serverové části. V případě klienta musí dojít ke schvalovacímu procesu na straně Applu. Pro úspěšné schválení a nasazení do App Store je potřeba splnit řadu podmínek, které zaměstnanci Applu kontrolují. V případě schválení je aplikace úspěšně zveřejněna v App Store.

9.1 Server

Pro nasazení serveru byl vybrán hosting, který umožnuje neomezenou kontrolu nad virtuálním strojem. Vybraným operačním systémem se stalo Ubuntu ve verzi 14.04. Díky neomezené kontrole je možné na takto zvoleném systému zprovoznit vše potřebné pro spuštění a běh serverové části.

Nejprve je potřeba zprovoznit prostředí podporující Java Enterprise Edition. Pro produkční server je použito prostředí GlassFish 4.0 a to hlavně, protože stejné prostředí bylo použito i při testování. K nainstalování serveru GlassFish je nutné stáhnout instalační balíček s danou verzí. Po stažení je potřeba balíček nainstalovat. Před začátkem nasazování serverové části je potřeba spustit doménu a databázi. S instalací, zprovozněním domény a databáze pomůže dokumentace, která je uveřejněná na oficiálních stránkách projektu GlassFish[6].

Po zprovoznění prostředí GlassFish a spuštění databáze je možno přejít přímo k nasazení samotného projektu. Pro nasazení je nutno zkopírovat na server soubor **PhantomAPI.war**[7]. Po zkopírování souboru je potřeba do terminálu zadat příkaz pro nasazení. Následující příkaz znázorňuje nasazení balíčku serverové části:

- asadmin deploy /cesta_k_souboru[8]/PhantomAPI.war

Po zadání příkazu dojde k nasazení balíčku s projektem a následnému spuštění. V případě úspěšného nasazení bude na následujícím příkladu URL adresy spuštěna serverová část:

- http://ip_adresa_serveru[9]:8080/PhantomAPI/

9.2 Klient

Před nasazením klienta je potřeba vytvořit unikátní identifikátor aplikace a distribuční certifikáty. Unikátní identifikátor slouží k jednoznačné identifikaci aplikace. Distribuční certifikáty poté slouží k tomu, aby daná aplikace mohla být nahrána ke schválení. Obě

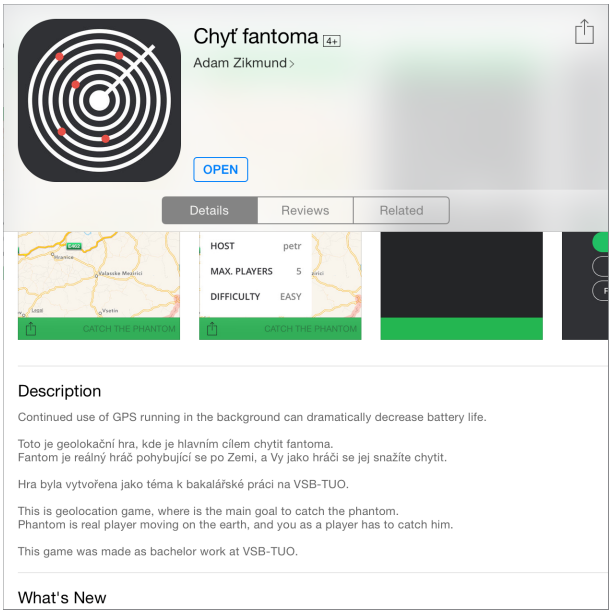
[6] Oficiální webové stránky projektu GlassFish https://glassfish.java.net/
[7] Uvedený soubor PhantomAPI.war se nachází na přiloženém CD
[8] Danou část je potřeba nahradit přesnou cestou k umístění souboru
[9] Danou část je potřeba nahradit IP adresou serveru na kterém je spuštěn GlassFish

tyto prerekvizity je nutno vygenerovat v administraci vývojářského centra pro iOS, ke kterému mají přístup pouze vývojáři mající předplacený iOS Developer Program.

Následně po vygenerování, je nutno, vytvořit novou aplikaci ve službě iTunes Connect[10]. Služba iTunes Connect se stará o správu aplikací, přehled stažení jednotlivých aplikací, platební informace vývojáře a jiné. K této službě mají rovněž přístup pouze předplatitelé. Při vytváření nové aplikace v této službě je potřeba zadat identifikátor aplikace z předešlého kroku. Dále je nutno vyplnit název, verzi a lokalizaci aplikace. Po dokončení tohoto kroku bude aplikace úspěšně vytvořena a zobrazena v nabídce aplikací.

Po vytvoření nové aplikace, je potřeba, vyplnit všechny požadované informace. Po vyplnění informací, je potřeba, nahrát snímky obrazovky a ikonu aplikace. Posledním krokem před odesláním aplikace ke schválení, je nahrání aplikace z vývojového prostředí Xcode. Pokud nahrání aplikace proběhlo v pořádku, zobrazí se v iTunes Connect informace o nahrané verzi. Následně již stačí nahranou verzi aplikace přiřadit a aplikace může být odeslána ke schválení. Schvalovací proces poté probíhá přibližně sedm pracovních dní.

Aplikaci je možno po schválení publikovat do App Store. Po publikování je poté aplikace zobrazena v App Storu a lze ji vyhledat pomocí příslušné kategorie či názvu. Aplikace poté může být stažena z App Store na libovolné zařízení, které podporuje iOS ve verzi 8[11].



Obrázek 5: Snímek obrazovky zobrazující hru Catch the phantom v App Store

[10] Webová stránka služby iTunes Connect https://itunesconnect.apple.com/

[11] Odkaz ke stažení hry Catch the phantom z App Store https://itunes.apple.com/us/app/id974985376?mt=8

10 Závěr

V práci bylo dosaženo hlavního cíle a to vytvoření geolokační hry Catch the Phantom. K tomu dopomohlo využití nástrojů pro analýzu a návrh a také použití moderních technologií při implementaci. Úspěšně také proběhlo samotné testování a nasazení hry do reálného provozu.

Jedním z přínosu práce je, získání všeobecného přehledu v oblasti mobilních technologií. Dále zlepšení analytického myšlení při analýze zadání. Také došlo k získání nových zkušeností s technologiemi použitými při návrhu a následné implementaci. Závěrečným přínosem je také zkušenost se schvalovacím procesem aplikace do App Store.

Hru je nadále možno rozšiřovat a vylepšovat. Možným rozšířením hry je, přidání statistiky pro jednotlivé hráče. Tímto rozšířením získají hráči lepší přehled o odehraných hrách a také si mohou zobrazit statistiky jiných hráčů. Dobrým příkladem pro rozšíření je také přidání možnosti komunikace mezi jednotlivými hráči. Přidáním komunikace se nabízejí možnosti domluvy a případné plánování strategie hráči během hry. Jako další rozšíření se nabízí možnost přidání podpory pro sociální sítě jako je Facebook či Twitter.

Adam Zikmund

11 Reference

[1] Smartphone OS Market Share, Q4 2014. [online]. [cit. 2015-04-22]. Dostupné z: http://www.idc.com/prodserv/smartphone-os-market-share.jsp

[2] iOS versus Android. Apple App Store versus Google Play: Here comes the next battle in the app wars. [online]. [cit. 2015-04-22]. Dostupné z: http://www.zdnet.com/article/ios-versus-android-apple-app-store-versus-google-play-here-comes-the-next-battle-in-the-app-wars/

[3] Apple iPhone 1st-gen specs - Engadget. [online]. [cit. 2015-04-22]. Dostupné z: http://www.engadget.com/products/apple/iphone/1st-gen/specs/

[4] Total iPhone sales hit 700 million. [online]. [cit. 2015-04-22]. Dostupné z: http://www.uswitch.com/mobiles/news/2015/03/total_iphone_sales_hit_700_million/

[5] Apple Unveils iPod touch. [online]. [cit. 2015-04-22]. Dostupné z: https://www.apple.com/pr/library/2007/09/05Apple-Unveils-iPod-touch.html

[6] Apple Launches iPad. [online]. [cit. 2015-04-22]. Dostupné z: https://www.apple.com/pr/library/2010/01/27Apple-Launches-iPad.html

[7] Populární hru Ingress si nově zahrajete i ve vybraných nákupních centrech. [online]. [cit. 2015-04-22]. Dostupné z: http://smartmania.cz/bleskovky/popularni-hru-ingress-si-nove-zahrajete-i-ve-vybranych-nakupnich-centrech-8085

[8] Top 5 iPhone Apps For Geocaching. [online]. [cit. 2015-04-22]. Dostupné z: http://www.notaboutthenumbers.com/2012/09/03/top-5-iphone-apps-for-geocaching/

[9] Traveller's Quest in Review – Geocaching lite!. [online]. [cit. 2015-04-22]. Dostupné z: http://www.touchmyapps.com/2010/01/20/traveller%E2%80%99s-quest-in-review/

[10] CodeRunner turns players into virtual spies in the real world. [online]. [cit. 2015-04-22]. Dostupné z: http://www.adweek.com/socialtimes/coderunner-turns-players-into-virtual-spies-in-the-real- world/521455

[11] Swarm by Foursquare Review. [online]. [cit. 2015-04-22]. Dostupné z: http://www.148apps.com/reviews/swarm-foursquare-review/

[12] Čtenáři doporučují: 19 aplikací na jarní výlet. [online]. [cit. 2015-04-22]. Dostupné z: http://www.svetandroida.cz/ctenari-doporucuji-19-aplikaci-jarni-vylet-201404

[13] Stručná historie Objective-C. [online]. [cit. 2015-04-22]. Dostupné z: http://babel.blog.root.cz/2011/09/20/strucna-historie-objective-c/

[14] History of Java programming language. [online]. [cit. 2015-04-22]. Dostupné z: http://www.freejavaguide.com/history.html

[15] Historie SQL. [online]. [cit. 2015-04-22]. Dostupné z: http://www.fi.muni.cz/usr/jkucera/pv109/2004/xbukovsk.htm

[16] Stopařův průvodce REST API. [online]. [cit. 2015-04-22]. Dostupné z: http://www.itnetwork.cz/stoparuv-pruvodce-rest-api

[17] JSON : jednotný formát pro výměnu dat. [online]. [cit. 2015-04-22]. Dostupné z: http://www.zdrojak.cz/clanky/json-jednotny-format-pro-vymenu-dat/

A Obsah přiloženého CD

Na přiloženém CD se nacházejí tyto soubory a složky:

- ZdrojovéKódy – složka obsahuje všechny zdrojové kódy a knihovny použité při implementaci
- PráceLaTeX – složka obsahuje projekt s prací v LaTeXu
- PhantomAPI.war – war archív obsahuje sestavený projekt PhantomAPI